REMARKS

This Amendment and Response is filed in response to the Office Action mailed on September 20, 2007. Please amend the above-identified patent application accordingly.

Claims 1, 15, and 22 are amended herein, claims 2, 16, and 23 are canceled, and no claims are newly added; as a result, claims 1, 3-7, 15, 17-22, and 24-28 are pending in this application.

§103 Rejection of the Claims

Claims 1-3, 5, 7, 15-17, 19, 21-24, 26, and 28 were rejected under 35 U.S.C. § 103(a) as being unpatentable over Lehman, et al. (U.S. 4,796,179, hereinafter, "Lehman") in view of Gauthier et al., "Automatic Generation and Targeting of Application Specific Operating Systems and Embedded Systems Software", 2001, IEEE (hereinafter 'Gauthier').

Claims 4, 18, and 25 were rejected under 35 U.S.C. § 103(a) as being unpatentable over Lehman, in view of Gauthier and further in view of Xu et al., "On Satisfying Timing Constraints in Hard-Real-Time Systems", 1991, ACM (hereinafter 'Xu').

Claims 6, 20, and 27 were rejected under 35 U.S.C. § 103(a) as being unpatentable over Lehman, in view of Gauthier and Xu, and further in view of D. Lake (US 2004/0045003A1) (hereinafter 'Lake').

Applicant respectfully submits that the Office Action did not make out a *prima facie* case of obviousness, because even if combined, the cited references fail to teach or suggest all of the claim limitations of the independent claims of the present Application.

Applicant continues to assert that the previously submitted claims with the recited limitations distinguish over Lehman and all other art of record. However, the claims have been amended herein to further distinguish the claimed invention from the art of record. Specifically, amended Claim 1 recites:

> A method for developing a real-time operating system, comprising:
> specifying a set of n tasks, task(1) through task(n), to be scheduled for execution;
> specifying t init-tasks that are executed only once upon initial execution of a task scheduler, t being less than or equal to n;

specifying a scheduling algorithm for scheduling the execution of said set of n tasks; and

synthesizing source code from commands embedded in source code to implement ~~a~~ the task scheduler that uses said scheduling algorithm for controlling execution of said set of n tasks, the task scheduler further controlling one execution of each of said set of t init-tasks, said synthesized source code being executable on a target system after compilation.

(Emphasis Added)

As correctly admitted in the Office Action at pgs. 3 and 4, "Lehman does not explicitly disclose ... synthesizing source code from commands embedded in source code to implement a task scheduler that uses the scheduling algorithm for controlling execution of said n tasks, said synthesized source code being executable on a target system after compilation." Lehman does not use embedded commands. Rather, Lehman provides functional blocks where each functional block has a corresponding source code template which is used to generate the source code. Each template in Lehman includes invariant code which defines a computation, and variables for tailoring the source code to use any specified parameters and also to couple the source code to the memory locations for its inputs and outputs. The present invention as currently claimed does not use or need templates or invariant code. As such, Lehman fails to teach each and every element of claim 1.

Further, Lehman does not disclose the claimed, "specifying t init-tasks that are executed only once upon initial execution of a task scheduler, t being less than or equal to n." In the Office Action, it is asserted that Lehman at col. 3, lines 36-39; col. 9, lines 56-61, and col. 32, lines 44-47 teach this claim element. These cited portions of Lehman are set forth below.

> A code generation software module generates software for use in the specified control system. The software routines generated include: a subsystem software routine for each subsystem for performing the computations specified in the catalog database for that subsystem; a scheduler program for initiating the execution of the subsystem software routines in accordance with the repetition rate and skew specified for each corresponding subsystem; and ... (Lehman at col. 3, lines 30-39).

> Preemptive Priority-Based Periodic Scheduling. In the present

> invention, the code segments associated with each subsystem (i.e., each specified computational rate) must be executed on a periodic basis or in response to a trigger condition. A software routine generated by the present invention, called the Scheduler, controls not only when the execution of each code segment is initiated, but also which code segments should be executed first when two or more code segments are running concurrently. (Lehman at col. 9, lines 52-61).
>
> The Scheduler 260 is the heart of the generated code, since it includes the software interface between the subsystems and controls the initiation of execution of the subsystem tasks. (Lehman at col. 32, lines 44-47).

It appears that the examiner has misunderstood the phrase "initiation of execution" in the cited portions of Lehman. It is clear that Lehman is referring to beginning the execution of a task with a "repetition rate and skew specified for each corresponding subsystem," which is clearly different than an "init task" as presently claimed that is executed only once. As described above in the cited portions of Lehman, a scheduler is described that controls the execution of a set of code segments (also called subsystem tasks). However, Lehman does not describe or suggest specifying two types of tasks, one type that can be scheduled for execution using a specified scheduling algorithm, and another type of task that is executed only once upon the initial execution of the task scheduler. In Lehman, there is no distinction between types of subsystem tasks. As such, Lehman does not teach or suggest the combination of elements presently claimed in the independent claims presented herein.

Gauthier describes a method for the automatic generation of application specific operating systems and automatic targeting of application software. At section 3.5.3 of Gauthier, the reference states, "Code Expander takes as input a list of macro code from Code Selector and parameters (processor and allocation information[]) from Architecture Analyzer. It generates the final OS code by expanding the macro codes of elements to source codes (in C or assembly)." However, Gauthier also does not describe or suggest an implementation as presently claimed for specifying two types of tasks, one type that can be scheduled for execution using a specified scheduling algorithm, and another type of

task that is executed only once upon the initial execution of the task scheduler. Similarly, neither Xu nor Lake describe or suggest the combination of elements as presently claimed.

Therefore, Applicants respectfully submit that at least for the reasons set forth above, independent claims 1, 15, and 22 and their dependent claims are allowable over Lehman, Gauthier, Xu, Lake, and combinations therof. Applicant respectfully submits that the current rejections should be withdrawn.

CONCLUSION

Applicant respectfully submits that the claims are in condition for allowance, and notification to that effect is earnestly requested. The Examiner is invited to telephone Applicant's attorney, Jim H. Salter at 408-406-4855 to facilitate prosecution of this application.

Respectfully submitted,

ROBERT M. ZEIDMAN

By his Representatives,

Salter Intellectual Property Law
105 Thoreau Lane
Folsom, CA 95630
408-406-4855

Date Oct. 25, 2007

By _____
Jim H. Salter
Reg. No. 35,668

CERTIFICATE UNDER 37 CFR 1.8: The undersigned hereby certifies that this correspondence is being deposited with the United States Postal Service with sufficient postage as first class mail, in an envelope addressed to: Mail Stop Amendment, Commissioner of Patents, P.O. Box 1450, Alexandria, VA 22313-1450 on this 25th day of October, 2007.

Jim H. Salter _____

Name Signature